Learn Drools:Part III-(Filter Facts)

In the previous article, we see how Drools creates Cartesian product between Facts if there is an HAS-A relationship between Domain Object/Facts.

In this article we will learn How to Filter the Facts, You can think it as of using Where clause in SQL to filter the resultset.

Let understand it with the same problem we have used in the Previous article.

In previous article, we create the following Drools Rule.

rule "print cross product"

when

emp: Employee();

dept: Department();

then

System.out.println(dept.getName() + "::" + emp.getName()+ ": "+emp.getDept().getName());

End

We insert two Department Objects and two Employee Objects, So when rule fires it creates all possible combination of employee and department that is four different combinations.

Civil::Shamik Mitra: IT

IT::Shamik Mitra: IT

Civil::Samir Mitra: Civil

IT::Samir Mitra: Civil

Obviously, this is not the optimal query we can do better than that, we want to filter in such a ways so it will fetch only those combinations where department name matches with Employees department name.

In SQL it is just a piece of cake,

***Select \* from Employee emp Department dept where emp.deptId=dept.deptId and dept.name=’?’***

***Where ? is a placeholder runtime we pass the value as IT or Civil***

**Now How can we do it in Drools Rule?**

**Believe me, it is also a piece of cake, I will do some tweaks in above rules to filter the combination.**

rule "filter cross product"

when

$dept: Department();

$emp: Employee(dept == $dept);

then

System.out.println($dept.getName() + "::" + $emp.getName()+ ": "+$emp.getDept().getName());

end

Please pay attention to the rule , here in When part we hold the incoming Department domain object/fact in $dept reference, then check is this Department same with Employees department?

If so we trigger then part else do nothing.

So if there is two Department Objects come as facts, rules fire on both department objects and check it with Employee's department association and only fetch valid combinations.

**Output:**

Civil::Samir Mitra: Civil

IT::Shamik Mitra: IT

Notice that it only fetch the valid combinations, not all the combinations.

**Quiz Time**

**We want to give a laptop to those people who are in IT department and designation is the manager?**

How you write the Drools rule?

Before seeing the answer take a paper and try to write the solution.

**Solution**

rule "give Laptop"

when

$dept: Department(name=="IT");

$emp: Employee(dept == $dept,manager==true);

then

$emp.setMessage("Give Laptop");

System.out.println($emp.getName()+ ": "+$emp.getDept().getName()+ ":"+$emp.getMessage());

end

Here in When part we want to fetch only those Department facts or Objects whose department name property equals to ‘IT’ and then try to find those Employees facts/Objects whose department match with IT and manager property is true, Note that we use “,” in Employee (dept == $dept,manager==true) by “,” we denotes a AND operation in Drools.

Equivalent SQL will be

***Select \* from Employee emp Department dept where emp.deptId=dept.deptId and dept.name=’IT’ and emp.manager=’true’***

**Output:**

Shamik Mitra: IT:Give Laptop

**Full Solution:**

**Employee.drl**

package com.rules

import com.example.droolsExample.pojo.Employee

import com.example.droolsExample.pojo.Department

rule "print cross product"

when

emp: Employee();

dept: Department();

then

//System.out.println("Fire print cross product Rule");

System.out.println(dept.getName() + "::" + emp.getName()+ ": "+emp.getDept().getName());

end

rule "filter cross product"

when

$dept: Department();

$emp: Employee(dept == $dept);

then

System.out.println($dept.getName() + "::" + $emp.getName()+ ": "+$emp.getDept().getName());

end

rule "give Laptop"

when

$dept: Department(name=="IT");

$emp: Employee(dept == $dept,manager==true);

then

$emp.setMessage("Give Laptop");

System.out.println($emp.getName()+ ": "+$emp.getDept().getName()+ ":"+$emp.getMessage());

end

**Employee.java**

package com.example.droolsExample.pojo;

public class Employee {

String name;

boolean manager;

String message;

Department dept;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public boolean isManager() {

return manager;

}

public void setManager(boolean manager) {

this.manager = manager;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public Department getDept() {

return dept;

}

public void setDept(Department dept) {

this.dept = dept;

}

}

**Department.java**

package com.example.droolsExample.pojo;

public class Department {

String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**DroolTest.java**

package com.example.droolsExample;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.Reader;

import org.drools.compiler.compiler.DroolsParserException;

import org.drools.compiler.compiler.PackageBuilder;

import org.drools.core.RuleBase;

import org.drools.core.RuleBaseFactory;

import org.drools.core.WorkingMemory;

import com.example.droolsExample.pojo.Department;

import com.example.droolsExample.pojo.Employee;

public class DroolsTest {

public static void main(String[] args) throws DroolsParserException,

IOException {

DroolsTest droolsTest = new DroolsTest();

droolsTest.executeDroolsEmployee();

}

public void executeDroolsEmployee() throws DroolsParserException, IOException {

PackageBuilder packageBuilder = new PackageBuilder();

String ruleFile = "/com/rules/employee.drl";

InputStream resourceAsStream = getClass().getResourceAsStream(ruleFile);

Reader reader = new InputStreamReader(resourceAsStream);

packageBuilder.addPackageFromDrl(reader);

org.drools.core.rule.Package rulesPackage = packageBuilder.getPackage();

RuleBase ruleBase = RuleBaseFactory.newRuleBase();

ruleBase.addPackage(rulesPackage);

WorkingMemory workingMemory = ruleBase.newStatefulSession();

Department dep = new Department();

dep.setName("Civil");

Department dep1 = new Department();

dep1.setName("IT");

Employee emp = new Employee();

emp.setName("Shamik Mitra");

emp.setManager(true);

emp.setDept(dep1);

Employee emp1 = new Employee();

emp1.setName("Samir Mitra");

emp1.setManager(true);

emp1.setDept(dep);

workingMemory.insert(dep);

workingMemory.insert(dep1);

workingMemory.insert(emp);

workingMemory.insert(emp1);

workingMemory.fireAllRules();

}

}

**Output:**

**Cross product:**

Civil::Shamik Mitra: IT

IT::Shamik Mitra: IT

Civil::Samir Mitra: Civil

IT::Samir Mitra: Civil

**Filter Cross Product**

Civil::Samir Mitra: Civil

IT::Shamik Mitra: IT

**Give laptop**

Shamik Mitra: IT:Give Laptop

In the next article we will discuss about Drools Inference.